Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Student Name

***Sanatkumar Rajmogali Ippalpalli***

Title of Project Report

***Guided Project 14 – Human Activity Recognition from Smart Phone Data***

Table of Contents

[Introduction 4](#_Toc83403205)

[Eckovation theme & Question 5](#_Toc83403206)

[Prerequisites before starting coding 6](#_Toc83403207)

[program DEVELOPMENT steps 8](#_Toc83403208)

[Dataset/Image requirements 8](#_Toc83403209)

[Technique – Supervised machine learning algorithm ANALYSIS 8](#_Toc83403210)

[PROGRAM / CODE DEVELOPMENT 9](#_Toc83403211)

[ANALYSIS 42](#_Toc83403212)

[CONCLUSION 43](#_Toc83403213)

[Figure 1 Import libraries and datasets/modules 10](#_Toc83403214)

[Figure 2 Load Pandas Dataset 11](#_Toc83403215)

[Figure 3 Visualization of Dataset 13](#_Toc83403216)

[Figure 4 t-SNE Method to find optimum clusters 14](#_Toc83403217)

[Figure 5 lightGBM Classification algorithm on Dataset 14](#_Toc83403218)

[Figure 6 Exploratory Data Analysis for the dataset 18](#_Toc83403219)

[Figure 7 Matplot lib plots for individual pattern 20](#_Toc83403220)

[Figure 8 Matplot lib plots using T-SNE 31](#_Toc83403221)

[Figure 9 Matplot lib plots 31](#_Toc83403222)

[Figure 10 Logistic Regression ML Algorithm 33](#_Toc83403223)

[Figure 11 Logistic Regression ML Algorithm 1 37](#_Toc83403224)

[Figure 12 SVC Classifier ML Algorithm 39](#_Toc83403225)

[Figure 13 Decision Tree Classifier ML Algorithm 40](#_Toc83403226)

[Figure 14 Random Forest Classifier ML Algorithm 42](#_Toc83403227)

***EXECUTIVE SUMMARY***

Science and technology improved many technologies and has guided numerous innovative features which recognize activities and can be clustered together. Each clustering to be identified and produce unerring accuracy for predictions.

As part of guided project, one of the important learning domains is decision tree / logistic regression along with Exploratory Data Analysis (EDA)used for the human activity’s recognition from smart phone dataset to produce faultless certainty in the real-world datasets.

Linear Discriminant Analysis is a well-known scheme for

feature extraction and dimension reduction

Linear Discriminant Analysis is a well-known scheme for

feature extraction and dimension reduction

Linear Discriminant Analysis is a well-known scheme for

feature extraction and dimension reduction

# Introduction

Science and technology improved many technologies and has guided numerous innovative features which advanced the techniques in data classification & clustering in unsupervised and supervised learning domain.

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

Y = f(X)

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

Supervised learning problems can be further grouped into regression and classification problems.

* **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
* **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

* Linear regression for regression problems.
* Random forest for classification and regression problems.
* Support vector machines for classification problems.

Hence Eckovation includes this guided project in the courseware for students to understand, implementation / execute the code themselves.

This report includes the 5W1H about the theme of development of code and running the code with database available over the internet. At the end of the report, the conclusions share the supervised machine learning model algorithms accuracy score.

# Eckovation theme & Question

**Theme : Human Activity Recognition from Smart Phone Data**

Recognizing human activities from temporal streams of sensory data observations is a very important task on a wide variety of applications in context recognition. Human activities are hierarchical in nature, i.e. the complex activities can be decomposed to several simpler ones. Human activity recognition is the problem of classifying sequences of accelerometer data recorded by pre-installed sensors in smart phones into known well-defined movements to make it ready for predictive modelling.

**Question:**

Perform activity recognition on the dataset using a hidden Markov model. Then perform the same task using a different classification algorithm (logistic regression/decision tree) of your choice and compare the performance of the two algorithms.

**Dataset Link: Human Activity Recognition with Smartphones**

<https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones>

# Prerequisites before starting coding

1. Who - Software needed?
2. What - Version / Release of software?
3. Any Prerequisites
4. How - to install the software
5. Which -libraries are needed to execute the problem statement
6. Where – dataset requirements, path location to include in the code
7. When – to use the above feature extraction
8. Who – Software neeed?

Python

1. What- Version / Release of software?

Python version 3.6 (latest version of python)

1. Any Prerequisites

RAM space availability & hard disk space availability

Admin rights to install the software

1. How - to install the software
2. The following url <https://www.python.org/downloads/>can be referred to download python.
3. Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url <https://www.anaconda.com/download/>
4. Which -libraries are needed to execute the problem statement
5. Sklearn dataset
6. Sklearn mixture
7. Sklearn preprocessing
8. Sklearn metrics
9. Sklearn cluster
10. Numpy (pip install numpy)
11. Matplotlib (pip install matplotlib)
12. Pandas
13. Seaborn
14. lightGBM
15. Cycler
16. Scipy
17. Collections
18. Where – dataset requirements, path location to include in the code
19. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: [https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-external-](https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/) [command/](https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/).
20. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.
21. When – to use the above feature extraction
22. When – to use the above technique

It is used for grouping of similar objects/variables data points together, based on their attributes or features and estimating classification with hyper tuning if needed.

It’s time to dive into the code!

# program DEVELOPMENT steps

* Dataset file
* Technique selections
* Program / code development
* Analysis

### Dataset/Image requirements

The source file used for this project are downloaded / collected from Kaggle website as instructed in the question.

**Dataset Link: Human Activity Recognition with Smartphones**

<https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones>

### Technique – Supervised machine learning algorithm ANALYSIS

For performing a Supervised Machine Learning Algorithm Model, following simple steps to be implemented.

1. Load the dataset in to the pandas dataframe
2. Plot the dataset using Matplot lib to understand the data distribution
3. Use Linear Regression Method
4. Use Random Forest Decision Tree Method
5. Compare the results & state analysis results

Let us hop to the inscribing carving!

### PROGRAM / CODE DEVELOPMENT

As explained step by step during the lecture by mentor, we would approach steps and understand the basics with brief explanation as needed.

#### Step 1: Import the relevant libraries and applicable datasets/modules

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Figure Import libraries and datasets/modules

Installation of relevant libraries is undertaken first. To remove the time taken for programming, these codes were made # (commented).

#### Step 2: Load dataset/Image

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Figure Load Pandas Dataset

Visulization of Dataset

# Group and count main names of columns

pd.DataFrame.from\_dict(Counter([col.split('-')[0].split('(')[0] for col in both\_df.columns]), orient='index').rename(columns={0:'count'}).sort\_values('count', ascending=False)

Graphical user interface, table

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Chart, bar chart

Description automatically generated

Figure Visualization of Dataset

#### Step 3: t-SNE Plot

Graphical user interface, text, application

Description automatically generated

Text, application

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Figure t-SNE Method to find optimum clusters

#### Step 4: lightGBM algorithm

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Figure lightGBM Classification algorithm on Dataset

Graphical user interface, text, application

Description automatically generated

Text, table

Description automatically generated with medium confidence

# Create duration datafrae

duration\_df = (both\_df.groupby([label, subject\_data])['Data'].count().reset\_index().groupby('Activity').agg({'Data':'mean'}) \* 1.28).rename(columns={'Data':'Seconds'})

activity\_df = pd.DataFrame(data, columns=['Activity', 'Accuracy']).set\_index('Activity')

activity\_df.join(duration\_df) Graphical user interface, table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

#### Step 5: Exploratory Data Analysis Algorithm

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Create dataframe and plot

sensor\_df = pd.DataFrame.from\_dict(data, orient='index').rename(columns={0:'Importance'})

sensor\_df.plot(kind='barh', figsize=(14,4), title='Sensor Importance For Classifing Participants By Walking Style (Feature Importance Sum)')

plt.show()

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Figure Exploratory Data Analysis for the dataset

Text

Description automatically generated

A picture containing bar chart

Description automatically generated

Graphical user interface, text

Description automatically generated

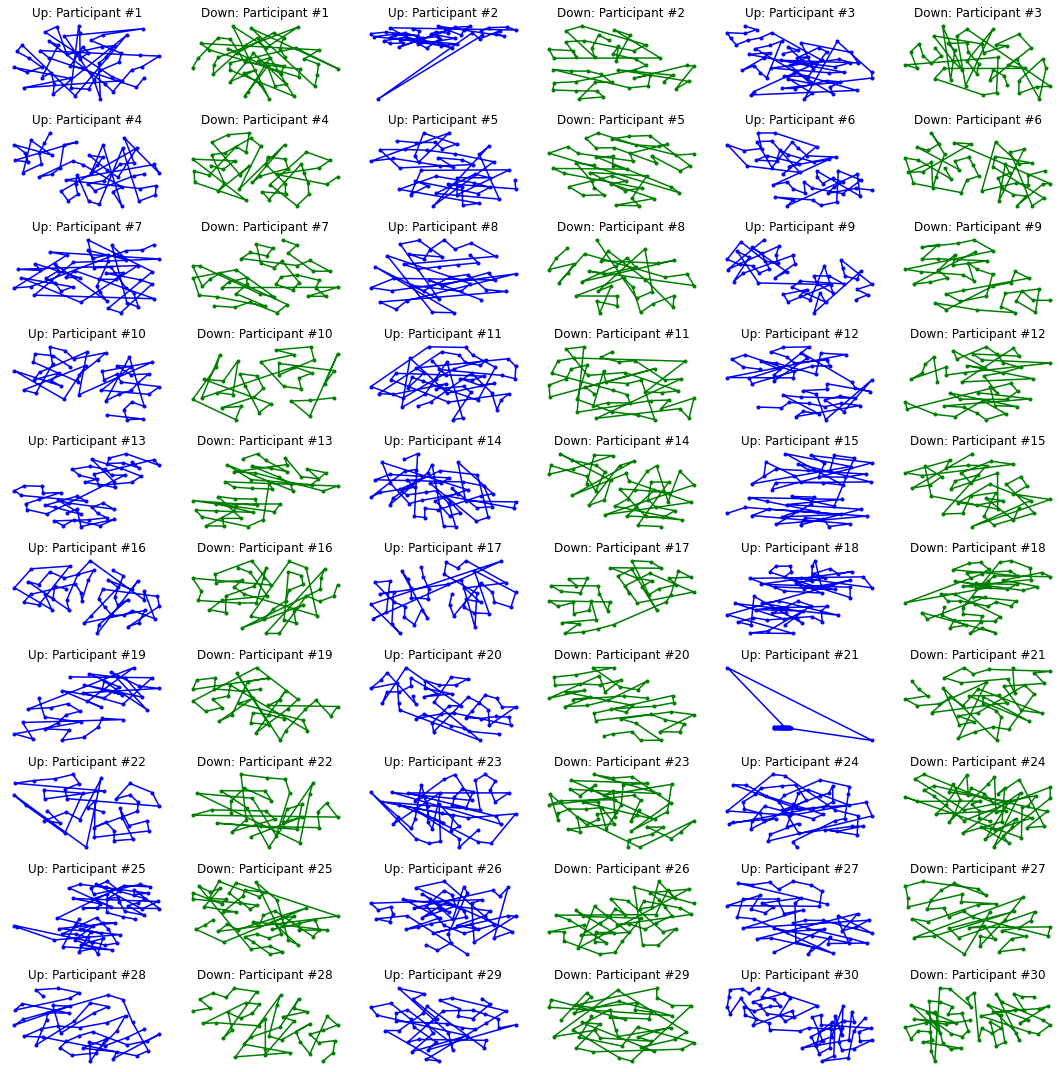


Figure Matplot lib plots for individual pattern

# Use SS class fro jdarcy

class SSA(object):

\_\_supported\_types = (pd.Series, np.ndarray, list)

def \_\_init\_\_(self, tseries, L, save\_mem=True):

'''

Decomposes the given time series with a singular-spectrum analysis. Assumes the values of the time series are

recorded at equal intervals.

Parameters

----------

tseries : The original time series, in the form of a Pandas Series, NumPy array or list.

L : The window length. Must be an integer 2 <= L <= N/2, where N is the length of the time series.

save\_mem : Conserve memory by not retaining the elementary matrices. Recommended for long time series with

thousands of values. Defaults to True.

Note: Even if an NumPy array or list is used for the initial time series, all time series returned will be

in the form of a Pandas Series or DataFrame object.

'''

# Tedious type-checking for the initial time series

if not isinstance(tseries, self.\_\_supported\_types):

raise TypeError('Unsupported time series object. Try Pandas Series, NumPy array or list.')

# Checks to save us from ourselves

self.N = len(tseries)

if not 2 <= L <= self.N/2:

raise ValueError('The window length must be in the interval [2, N/2].')

self.L = L

self.orig\_TS = pd.Series(tseries)

self.K = self.N - self.L + 1

# Embed the time series in a trajectory matrix

self.X = np.array([self.orig\_TS.values[i:L+i] for i in range(0, self.K)]).T

# Decompose the trajectory matrix

self.U, self.Sigma, VT = np.linalg.svd(self.X)

self.d = np.linalg.matrix\_rank(self.X)

self.TS\_comps = np.zeros((self.N, self.d))

if not save\_mem:

# Construct and save all the elementary matrices

self.X\_elem = np.array([ self.Sigma[i]\*np.outer(self.U[:,i], VT[i,:]) for i in range(self.d) ])

# Diagonally average the elementary matrices, store them as columns in array.

for i in range(self.d):

X\_rev = self.X\_elem[i, ::-1]

self.TS\_comps[:,i] = [X\_rev.diagonal(j).mean() for j in range(-X\_rev.shape[0]+1, X\_rev.shape[1])]

self.V = VT.T

else:

# Reconstruct the elementary matrices without storing them

for i in range(self.d):

X\_elem = self.Sigma[i]\*np.outer(self.U[:,i], VT[i,:])

X\_rev = X\_elem[::-1]

self.TS\_comps[:,i] = [X\_rev.diagonal(j).mean() for j in range(-X\_rev.shape[0]+1, X\_rev.shape[1])]

self.X\_elem = 'Re-run with save\_mem=False to retain the elementary matrices.'

# The V array may also be very large under these circumstances, so we won't keep it.

self.V = 'Re-run with save\_mem=False to retain the V matrix.'

# Calculate the w-correlation matrix.

self.calc\_wcorr()

def components\_to\_df(self, n=0):

'''

Returns all the time series components in a single Pandas DataFrame object.

'''

if n > 0:

n = min(n, self.d)

else:

n = self.d

# Create list of columns - call them F0, F1, F2, ...

cols = ['F{}'.format(i) for i in range(n)]

return pd.DataFrame(self.TS\_comps[:, :n], columns=cols, index=self.orig\_TS.index)

def reconstruct(self, indices):

'''

Reconstructs the time series from its elementary components, using the given indices. Returns a Pandas Series

object with the reconstructed time series.

Parameters

----------

indices: An integer, list of integers or slice(n,m) object, representing the elementary components to sum.

'''

if isinstance(indices, int): indices = [indices]

ts\_vals = self.TS\_comps[:,indices].sum(axis=1)

return pd.Series(ts\_vals, index=self.orig\_TS.index)

def calc\_wcorr(self):

'''

Calculates the w-correlation matrix for the time series.

'''

# Calculate the weights

w = np.array(list(np.arange(self.L)+1) + [self.L]\*(self.K-self.L-1) + list(np.arange(self.L)+1)[::-1])

def w\_inner(F\_i, F\_j):

return w.dot(F\_i\*F\_j)

# Calculated weighted norms, ||F\_i||\_w, then invert.

F\_wnorms = np.array([w\_inner(self.TS\_comps[:,i], self.TS\_comps[:,i]) for i in range(self.d)])

F\_wnorms = F\_wnorms\*\*-0.5

# Calculate Wcorr.

self.Wcorr = np.identity(self.d)

for i in range(self.d):

for j in range(i+1,self.d):

self.Wcorr[i,j] = abs(w\_inner(self.TS\_comps[:,i], self.TS\_comps[:,j]) \* F\_wnorms[i] \* F\_wnorms[j])

self.Wcorr[j,i] = self.Wcorr[i,j]

def plot\_wcorr(self, min=None, max=None):

'''

Plots the w-correlation matrix for the decomposed time series.

'''

if min is None:

min = 0

if max is None:

max = self.d

if self.Wcorr is None:

self.calc\_wcorr()

ax = plt.imshow(self.Wcorr)

plt.xlabel(r'$\tilde{F}\_i$')

plt.ylabel(r'$\tilde{F}\_j$')

plt.colorbar(ax.colorbar, fraction=0.045)

ax.colorbar.set\_label('$W\_{i,j}$')

plt.clim(0,1)

# For plotting purposes:

if max == self.d:

max\_rnge = self.d-1

else:

max\_rnge = max

plt.xlim(min-0.5, max\_rnge+0.5)

plt.ylim(max\_rnge+0.5, min-0.5)

# Euclidean norm of the acceleration

walking\_series = both\_df[(label=='WALKING') & (both\_df['subject']=='#1')][['tBodyAcc-mean()-X', 'tBodyAcc-mean()-Y', 'tBodyAcc-mean()-Z']].reset\_index(drop=True)

walking\_series = (walking\_series\*\*2).sum(axis=1)\*\*0.5 A picture containing graphical user interface

Description automatically generated

#### Step 6: Pattern & t-SNE Plot

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Text

Description automatically generated

# Function to fit a sinus

def fit\_sin(tt, yy):

'''Fit sin to the input time sequence, and return fitting parameters "amp", "omega", "phase", "offset", "freq", "period" and "fitfunc"'''

tt = np.array(tt)

yy = np.array(yy)

# Assume uniform spacing

ff = np.fft.fftfreq(len(tt), (tt[1]-tt[0]))

Fyy = abs(np.fft.fft(yy))

# Exclude the zero frequency "peak"

guess\_freq = abs(ff[np.argmax(Fyy[1:])+1])

guess\_amp = np.std(yy) \* 2.\*\*0.5

guess\_offset = np.mean(yy)

guess = np.array([guess\_amp, 2.\*np.pi\*guess\_freq, 0., guess\_offset])

# Sinus

def sinfunc(t, A, w, p, c):

return A \* np.sin(w\*t + p) + c

# Fit sinus

popt, pcov = curve\_fit(sinfunc, tt, yy, p0=guess)

A, w, p, c = popt

f = w/(2.\*pi)

fitfunc = lambda t: A \* np.sin(w\*t + p) + c

return {"amp": A, "omega": w, "phase": p, "offset": c, "freq": f, "period": 1./f, "fitfunc": fitfunc, "maxcov": np.max(pcov), "rawres": (guess,popt,pcov)}

# Get data

main\_style1 = style1\_ssa.reconstruct([0, 1])

tt1 = main\_style1.index

yy1 = main\_style1.values

tt\_res1 = np.arange(0, 48, 0.1)

# Fit data

res1 = fit\_sin(tt1, yy1)

# Get data

main\_style2 = style2\_ssa.reconstruct([0, 1])

tt2 = main\_style2.index

yy2 = main\_style2.values

tt\_res2 = np.arange(0, 48, 0.1)

# Fit data

res2 = fit\_sin(tt2, yy2)

# Plot data

fig, axarr = plt.subplots(1, 2, figsize=(15,5))

# Plot data

axarr[0].plot(tt1, yy1, "-ok", label='Data', linewidth=2)

axarr[0].plot(tt\_res1, res1['fitfunc'](tt\_res1), "r-", label='Fit', linewidth=2)

axarr[0].set\_title('Style 1 Walking Fit: {:.2f} Steps Per Second'.format((res1['omega']\*1.28)/(pi)))

axarr[0].legend(loc="best")

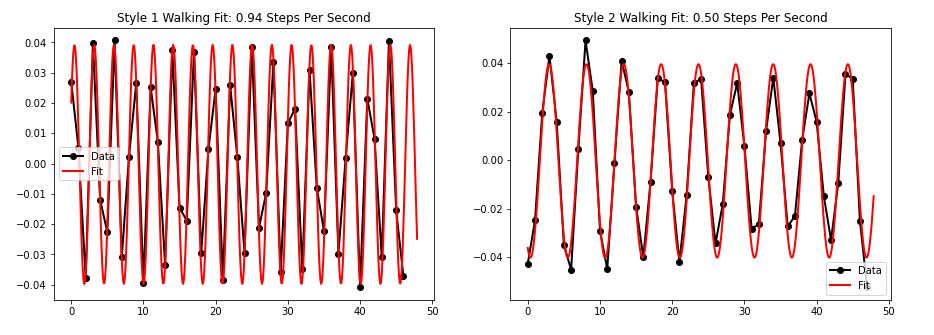
axarr[1].plot(tt2, yy2, "-ok", label='Data', linewidth=2)

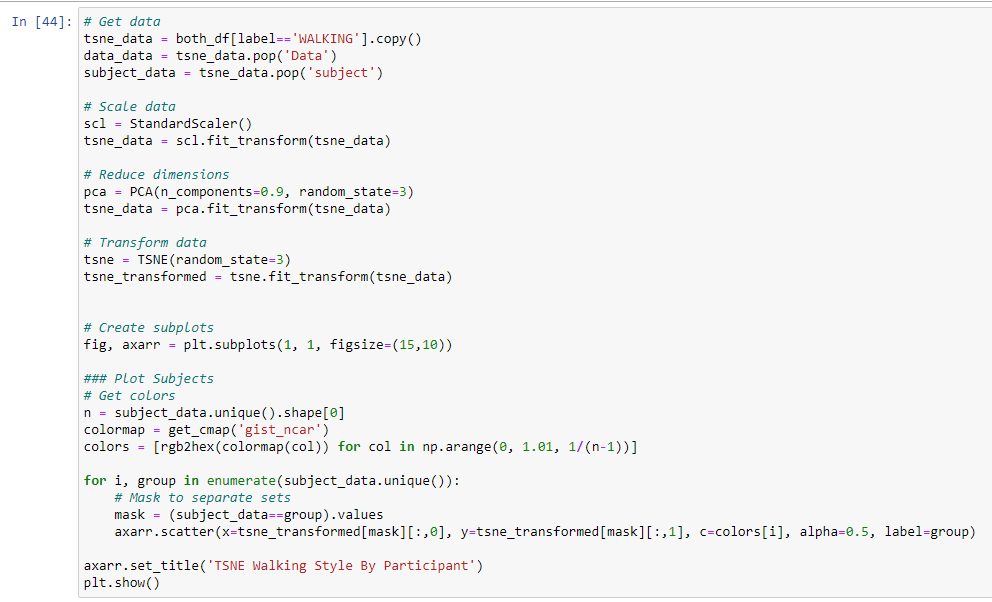
axarr[1].plot(tt\_res2, res2['fitfunc'](tt\_res2), "r-", label='Fit', linewidth=2)

axarr[1].set\_title('Style 2 Walking Fit: {:.2f} Steps Per Second'.format((res2['omega']\*1.28)/(pi)))

axarr[1].legend(loc="best")

plt.show()





Chart, scatter chart

Description automatically generated

Figure 8 Matplot lib plots using T-SNE

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Figure 9 Matplot lib plots

#### Step 7: Logistic Regression

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generated

Text

Description automatically generated

Figure 10 Logistic Regression ML Algorithm

Chart, bar chart

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

A picture containing chart

Description automatically generated

Text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Figure 11 Logistic Regression ML Algorithm 1

#### Step 8: Linear SVC (Support Vector Classifier)

Graphical user interface, text, application

Description automatically generated

Chart

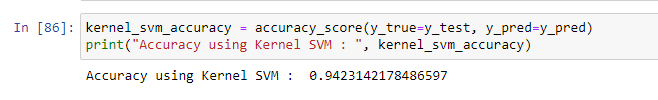
Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated



Chart

Description automatically generated

Text, letter

Description automatically generated

Figure 12 SVC Classifier ML Algorithm

#### Step 9: Decision Tree Classifier

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing chart

Description automatically generated

Text, letter

Description automatically generated

Figure 13 Decision Tree Classifier ML Algorithm

#### Step 10: Random Forest Classifier

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generatedChart

Description automatically generated with medium confidence

Text

Description automatically generated

Figure 14 Random Forest Classifier ML Algorithm

### ANALYSIS

Clustering algorithm is used on clustering dataset advised by Eckovation and executed all the classification techniques with estimation of accuracy score.

This entire program runs within few minutes to execute all codes for all machine learning algorithm.

# CONCLUSION

In this guided project, we built supervised and unsupervised machine learning models and predicted accuracy score.

The accuracy obtained through these models is as follows -

| Logistic | Linear SVM | Kernel SVM | Decision Trees | Random Forest |
| --- | --- | --- | --- | --- |
| 95.55 | 96.84 | 94.23 | 87.24 | 91.92 |

This entire program runs within few minutes.

references:

1. https://www.kaggle.com/morrisb/what-does-your-smartphone-know-about-you
2. <https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones/code?datasetId=226&sortBy=voteCount>
3. <https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>
4. <https://github.com/anas337/Human-Activity-Recognition-Using-Smartphones.github.io>